Limbo Champion in the Low Cost VR competition: The Power Glove Serial Interface

Ben Gross
Junior, Sociology
University of Illinois
PGSI Project Leader
b-gross@uiuc.edu

Jim Brain
Senior, Computer Engineering
University of Illinois
PGSI Designer
j-brain@uiuc.edu

Abstract

The Student Chapter of the Association for Computing Machinery at the University of Illinois at Urbana-Champaign developed the PowerGlove Serial Interface (PGSI) as the next step in low-cost virtual reality interface technology. Specifically, the Special Interest Group in Computer Architecture (SIGArch) developed the unit as a fundraising project to aid in the purchase of integrated circuits and test equipment to develop more projects like the PGSI. Ben Gross heads up the team that has worked on the project from early in February 1992 to the present. Jim Brain completed most of the PGSI development and early prototypes. Students were involved in completing nearly every step of the process from conception to shipping the finished product.

1 Introduction

For those not connected to a research institution or large corporation, the start up costs for VR experiments or studies can be and usually are prohibitive. Interested individuals all over the world are searching for low-cost alternative devices capable enough for beginning experiments. For some time, one such device has been the Mattel PowerGlove, manufactured for the Nintendo Entertainment System (NES). This device provides a low cost substitute for the more powerful Polhemus Fastrak and the VPL DataGlove combination. For many, this glove had the power necessary for early experiments that did not have a ten thousand dollar and up budget, but it lacked one important item, a standard interface. Since the glove was designed for the NES, it used a non-standard 7 pin connector, and the protocol is a modified synchronous serial transfer. This made it more difficult to implement in low-cost VR.

2 Initial Problems

During the initial design we encountered several problems, these included the PowerGlove having a non standard connector and the modified serial protocol that was it's output. The non-standard Nintendo connector became almost trivial when compared to the modified serial protocol, since the latter requires special software to perform the protocol emulation, whereas the former can be eliminated by replacing the connector. Although these problems plagued its widespread use, many interested individuals set out and developed workarounds to the protocol problem. Curtis Nintendo Super Extendo Cables, now manufactured by Nuby Manufacturing, allowed us to put a Nintendo connector directly on our interface so that the user would not have to decapitate his or her PowerGlove cable. Now that the protocol alone prevented operation of the glove, those who developed adapters took two related approaches to provide the solution.

Many early users of the glove hooked the unit up to appropriate input/output pins on personal computers, and wrote protocol emulation routines on the host system to poll the glove for input. Early examples include various parallel port hookups for the Amiga, IBM, and Atari ST computer systems. This approach has two flaws. First, the interface is not sold commercially, so users were expected to build the interface themselves, or have someone build it for them. Second, since correct operation depends on protocol conversion and polling, each CPU and/or operating system change required a new piece of software to perform the required actions. This inhibited development. Also, this second restriction usually only allows operation on computer/operating system combinations that allowed the user access to hardware input/output. This made operation on high powered workstations extremely difficult (but not impossible).

0-7803-1363-1/93 $3.00 © 1993 IEEE

3 A Short History

We saw a need for a standard interface between the PowerGlove and the host CPU. Examples of this second approach include the GoldBrick interface and the original PowerGlove Serial Adapter(PGSA) from AGE, Inc. These devices provided a level of abstraction between the user and the glove, and the PGSA allowed any computer system with a standard RS-232 port to utilize the glove. The GoldBrick interface was tailored to operate on the Apple Desktop Bus (ADB) found on Macintosh style computers, while the PGSA worked on standard RS-232 ports. This hardware approach is the category that the PowerGlove Serial Interface (PGSI) falls into. By providing a standard interface to the host CPU, the PGSI allows users to port code to different machines with a minimum of rewriting. The main fault of the PowerGlove Serial Adapter is that very few were ever produced. Although the design is still being manufactured, it is time for a new contender in the race to bring Virtual Reality devices to those who need them.

In January of 1992, Ben Gross, then a freshman at the University of Illinois in education finds an interesting request for help on a new interface from Greg Newby, an assistant professor in the Graduate School of Information Sciences. Newby, as a graduate student at Syracuse University, became interested in the work of the creators of the PGSA, AGE Inc. AGE, Inc., which stands for Abrams-Gentile Entertainment, had sold the rights to the design of the PowerGlove to Mattel Toys, Inc., and had designed an interface that would allow anyone with a standard RS-232 compliant computer to access the glove. When Newby took his post at the University of Illinois, he brought along the box and permission from AGE to make more if he wanted to. Greg, upon realizing the complexity of the device, shows Ben an original PGSA box after a virtual reality lecture and explains his contact with AGE. Ben then finds out that Greg has secured the OK to build more of these boxes, so Ben volunteers to bring up the idea at the next meeting of SIGArch, a special Interest Group at the University of Illinois Student Chapter of the Association for Computing Machinery.

At that meeting, Ben explained the project to an audience consisting of people such as Jim Brain, Mike Schaffstein, Mike Stangel (Chair of SIGArch at the time), Teresa Johnson, and others. Mike Stangel proposes that SIGArch work on this as a possible fund-raiser, to help SIGArch acquire equipment and supplies to work on other projects like this one. After initial meetings with Greg Newby, Mike appoints Ben Gross as project leader. Ben enlists the help of Jim Brain and Mike Schaffstein to tackle the problem of duplicating the little box which Greg has. After initial checks on prices and availability, Both Jim and Mike determine that the high price of a 68705 microcontroller, coupled with an incomplete listing of the code inside the controller, necessitated a new design. Since both Jim and Mike knew the inner design of Motorola 68HC11 microcontrollers, a decision was made to design a project around that chip. Along the way, Ben finds code written for just a design. It turns out that Ron Menelli designed an interface around a 68HC11 to act as a PowerGlove controller. The code is a rewrite of code revised by Dave Stampe for REND386 . The code has its origins in Germany. Jim secures the rights to use this code in a new product, called the PGSA II. Along the way, it became easier to develop a new interface rather than reverse-engineer the old one, and the PGSI was born.

4 Basic Description

The unit measures 1"x1.5"x3.35" and resembles a elongated gender changer, it houses a complete Motorola MC68HC11E2FN (HC11) imbedded microcontroller, associated support circuitry, and appropriate interface connectors. The PGSI connects to any personal computer or workstation with an EIA-232D standard (RS-232) serial communications port capable of 9600 bps at no parity and 1 stop bit (9600,N,8,1) available as a 25 pin DB-25 style connector. Adapters are available for users with a newer style IBM or IBM compatible systems that have a 9 pin DB-9 connector and users with Macintosh computers.
Since the unit employs a microprocessor to enable operation, the device has a rich set of commands and can be configured for a variety of uses. The heart of the system, an HC11 processor, has 2048 bytes of Electrically Erasable Programmable Read Only Memory (EEPROM) and 256 bytes of Random Access memory (RAM). The EEPROM allows the control program to be located on-chip, and the unique properties of the EEPROM allow end users to upgrade the control program themselves, without needing to ship the unit back for upgrading. The entire program fits in 2K of EEPROM, and 256 bytes of RAM is more than adequate for all temporary storage.

As with all microcontrollers, the HC11 has 40 various input/output pins. The Mattel PowerGlove connects up to the microcontroller via these I/O pins, and a control program stored in EEPROM (firmware) controls the state of those pins or reads their value at appropriate times. The end user sends a special one byte command to the PGSI through the serial port requesting PowerGlove information, and the firmware parses that command and returns a packet of information. This packet, sent to the host CPU via serial transfer, contains all the information available from the PowerGlove. The end user is completely unaware of the necessary protocol conversion and does not need to poll the glove when it is not needed. The polling command is only one of many that can be interpreted by the firmware. This makes the device very functional.

5 The Details

The PGSI contains 35 commands in the complete command set. This includes those commands required for complete AGE and Menelli emulation (differences are explained below), as well as superset commands found only in the PGSI. The complete command set is shown in Figure 1 and contains basic information about each command. To operate the PGSI, the user/programmer need only be concerned with commands 0x07, 0x01, 0x02, or 0x07, 0x43, 0x52, and 0x3F. The remainder of command sequences simply initiate extended features or access various filters.

To understand the complexity of the PGSI design and appreciate the relatively few commands needed to operate the unit, it is necessary to examine the features the PGSI provides. One of the most basic of features is the ability to emulate either the original AGE PGSA interface or the newer Menelli device. This means that the PGSI must not only transmit information in the correct sequence, but must also accept all commands for that emulation and switch emulation modes transparently. Information is transmitted in packets, which contain all pertinent information about the glove normally transmitted by the respective emulation mode. Initially, the AGE packet only contains bytes 00-10, whereas the Menelli packet contains bytes 00-05 with a pre-packet header byte, 0xA0, sent as the first byte in continuous mode as a delimiter.

In addition to providing two separate packet structures, the PGSI must relay information to the host CPU in one of two ways. The first, continuous mode, instructs the PGSI to send a new packet of information to the host CPU as soon as it is received from the glove. The second, request mode, instructs the PGSI to send a new packet upon reception of a send packet command. Packets are not stored, so the packet received is always the newest. This packet structure ensures that all relevant information about the glove is transmitted and provides the user/programmer with an easy way to gather data from the glove

Even though the PGSI provides two complete emulation modes, they are by no means mutually exclusive. Notice that most commands can be accessed from either mode. This allows users in one emulation to access the filter commands in either emulation. If a user attempts to access a command from one emulation mode that is only available in the other emulation mode, the firmware will automatically accept the command and switch emulations. This means a user can put the PGSI in Menelli request mode with 'R' and turn the rotation filter on with 0x08.

6 Operations

To request a packet of information from the PGSI, one must first determine which mode to use in a program. Programmers of new applications are strongly encouraged to use the AGE emulation mode, since this mode provides more complete information about the glove and contains more commands. Menelli mode is mainly intended for those applications previously written and for users wishing to access the PGSI using standard text commands.

Once the emulation mode is determined, the programmer must determine which type of data the program needs: continuous or request. Programs that track the hand in very tight loops benefit from the continuous output of data, while systems that only approach real time may benefit from the request mode. Currently, the glove will only update 27 times a second, so frequency of updates is a major concern.

The programmer now needs only include the function libraries from the PGSI developer's library, which includes all the basic driver routines as C-callable functions. If the programmer has decided on AGE request, just send 0x07, 0x02. This not only turns on request mode, but sends a packet of information, as

per the AGE emulation. The device will now send a new packet when instructed by sending the 0x02 command again. If the programmer needs AGE continuous mode, 0x07, 0x01 will put the PGSI in that mode. Menelli request is 0x07, 0x52, and the packet is accessed with 0x3F. Note that, unlike the AGE request option, Menelli request does not automatically send a packet when the device is put into request mode. This is due to command differences. AGE combined the features of 0x52 and 0x3F into one command, 0x02. Finally, Menelli continuous is 0x07, 0x43. These command sequences illustrate the relative ease of using this product to acquire data from the PowerGlove.

Command Code (in HEX)	**Command Code** (in ASCII)	**Description**	**Emulation Available in**
0x01	none	Place interface in continuous mode	AGE
0x02	none	Place interface in request mode/request packet	AGE
0x04	none	Resend default initialization packet to the PowerGlove	AGE/Menelli
0x05	none	not implemented	none
0x06	none	not implemented	none
0x07	none	Reset PowerGlove and PGSI to default configuration	AGE/Menelli
0x08	none	Turn rotation filter on	AGE/Menelli
0x09	none	Turn rotation filter off	AGE/Menelli
0x0A	none	Turn PowerGlove on for input	AGE/Menelli
0x0B	none	Turn PowerGlove off for input	AGE/Menelli
0x0C	none	not implemented	AGE/Menelli
0x0D	none	not implemented	AGE/Menelli
0x0E	none	Turn on digital inputs	AGE/Menelli
0x0F	none	Turn off digital inputs	AGE/Menelli
0x10	none	Turn on first 4 channels of A/D	AGE/Menelli
0x11	none	Turn off first 4 channels of A/D	AGE/Menelli
0x12	none	Turn on second 4 channels of A/D	AGE/Menelli
0x13	none	Turn off second 4 channels of A/D	AGE/Menelli
0x14	none	Turn left lens dark, leave right lens in current state	AGE/Menelli
0x15	none	Turn left lens clear, leave right lens in current state	AGE/Menelli
0x16	none	Turn right lens dark, leave left lens in current state	AGE/Menelli
0x17	none	Turn right lens clear, leave left lens in current state	AGE/Menelli
0x18	none	Turn both lenses dark	AGE/Menelli
0x19	none	Turn both lenses clear	AGE/Menelli
0x1A	none	Turn left lens dark and right lens clear	AGE/Menelli
0x1B	none	Turn left lens clear and right lens dark	AGE/Menelli
0x1C,0x01-0x08	none	Change baud rate to value specified after command byte	AGE/Menelli
0x1D	none	Change chopping frequency of LCD lens driver circuitry, range: 16Hz to 5kHz.	AGE/Menelli
0x1E,0x00-0xFF	none	Send digital data to output port (Second byte is value to send)	AGE/Menelli
0x2B	'+'	Turn hysterisis deglitching on	AGE/Menelli
0x2D	'-'	Turn hysterisis deglitching off	AGE/Menelli
0x3F	'?'	Request a packet in Menelli emulation	Menelli
0x43	'C'	Place interface in continuous mode	Menelli
0x52	'R'	Place interface in request mode	Menelli

Figure 1: The command set

8 Three Dimensional Viewing

When the PGSI was in early development, many people expressed an interest in having one device that could control both the PowerGlove and a pair of SEGA or Toshiba style 3-D shutter glasses, also known as field sequential vision glasses. These glasses have two LCD 'light valves', one for each eye, that control the passage of light into the respective eye. By using the glasses and a computer monitor, it is possible to create the illusion of stereoscopic vision. Although the technique is straightforward, it warrants a passing discussion here. First, draw two pictures, one of the object(s) as seen through only the left eye, and the other as seen through only the right eye. Next, display the left eye image on the computer monitor and let the left eye (and only the left eye) view it. Next display the right eye image and allow the right eye to view it. Repeat this display sequence (left,right) continuously and in rapid succession, and the human eye will, with the aid of the glasses, merge the two 2-D images into one 3-D image. This is stereoscopic vision. The PGSI will allow the connection of one set of 3-D glasses for the purpose of providing stereoscopic vision, and the glasses can be controlled through either firmware commands or by controlling specific I/O pins on the host CPU directly.

If the application requires or will benefit from use of the LCD shutter glasses, it can control the glasses in one of three ways. The first two involve direct hardware I/O accesses, while the third can control the glasses via serially transmitted commands. In the first mode, the DTR and RTS lines on the serial port are used to control each lens individually. The second mode allows flipping between lenses with just one of those lines, meaning that the user can have left lens on : right lens off or vice versa. This mode is intended for Macintosh users who wish to control the glasses via hardware, but have access to only one digital output pin. The third mode uses commands sent over the serial connection to control the lenses. This approach suffers slightly in speed comparisons, but can be used on all computer systems, regardless of hardware accessibility. This new functionality was added in the form of Glen Harris's SEGA driver circuitry. The device can control the glasses uses two output pins on the RS-232 port (Glen's design), control with one wire (Dave Stampe's and other's initial design), or no wires (Jim Brain design with software)
The driver circuitry is capable of the following:

One wire control: (LEFTON/RIGHTOFF, LEFTOFF/RIGHTON)
Two wire control: (LEFTON/RIGHTOFF, LEFTOFF/RIGHTON
LEFTOFF/RIGHTOFF, LEFTON, RIGHTON)
No wire control (Use software RS-232 commands to emulate two wire control)

Figure 2: LCD shutter glasses commands.

9 Applications and Software

There exist few software titles which use the PowerGlove with any adapter. Since the device is a derivative of the AGE PGSA (and was called the PGSA II while in early development), the PGSI includes full compatibility with production versions of the PGSA. Also, since the device borrows code and design from a home-brew adapter by Ron Menelli (available via FTP), complete compatibility with the Menelli design has been provided. Since the interface must juggle two complete emulation modes, a few assumptions have been made on the initial behavior of the calling program, like proper initialization of the device. Also, since both of these early designs did not include any provisions for more than one set of commands, the PGSI will autoset the emulation mode. It can do this easily because the command sets are completely different. Thus, certain commands imply a particular emulation mode.

Currently we have a functional Mac mouse driver replacement written by one of our members, Jay Kreibich (jak@uiuc.edu). We will include the binary and the source with the PGSI Mac distribution. Jay should also have libraries in the near future. We also have PC and windows versions of the drivers in the works. Another of our members Chris Wilson (cwilson@ncsa.uiuc.edu) has written a windows 3.1 application called Maestro II. Maestro allows you to use the glove on a virtual keyboard that outputs to a midi device.

There are a number of excellent packages that various authors have written to support the parallel version of the glove, such as Dave Stampe's Rend 386, that we hope will be ported to the PGSI as it becomes more widespread. If anyone is interested in writing drivers or other software we will gladly take submissions and distribute them. As of this date we have no plans to sell any of the software for the PGSI and will probably distribute anything we write under some sort of a GNU license. All of out available software will be placed on our FTP site.

10 Availability

There were a number of roadblocks that caused seemingly endless delays in our shipping the PGSIs. As with any student project the biggest problem is lack of resources, this includes time, money, and labor. After all the development and prototyping was finished, we solved our cash flow problems by taking advance orders, previous to this time frame was entirely another matter. Most of the initial supplies were paid for with various members credit cards and were not reimbursed until later. Economics lessons never come back so vividly as when one tries to organize college students into a small production facility. Another problem involved legalities. Generally, universities cannot sell items for profit, and the National Association for Computing Machinery is classified as a not-for-profit organization. Although ACM at UIUC stated that profits from the sale of the PGSI will be used for purchase of equipment and supplies for the Special Interest Group (SIG) developing the PGSI, legal problems caused several weeks worth of delays. Once we completed the circuit board design we sent it out to a company who bought the parts in bulk and mostly assembled the units for us. Even though we had to assemble a small part of the units, it took hours and hours to complete. In addition several parts were not in stock a had to be ordered causing further delays. Our final problem and one that caused the biggest delay was the power supply. The board had been designed with an uncommon power jack, and we could not find a large quantity of power supplies to fit them. Eventually, we removed the jacks from the power supplies and added out own. This was all compounded by the fact that much of this took place over the summer when most of the students, our labor force, had gone home.

11 Future Uses

There are many "extra" functions built into the PGSI. The HC11 contains eight 8-bit linear Analog-to-Digital converters, so the PGSI includes the capability to wire those inputs to any outside device and read the values through software. Although this has not been tested to a great extent, it is possible to use the A/D ports to read the finger flex positions with 8 bits of resolution per finger. The PGSI also has 8 digital inputs and 8 digital outputs that can be accessed through software.

12 Conclusion

This project provided an incredible amount of experience to all those involved. Our members learned about circuit design, printed circuit board etching, prototyping, marketing, production, and deadlines. This project has been more in depth than any our group has undertaken to date. Everyone involved came away with the kind of experience that is only available through hands on real life projects.

13 Contact Information

Email: (preferred method of contact)
pgsi@uiuc.edu
FTP:
ftp.cso.uiuc.edu in /ACM/PGSI

Paper Address:
The Student Chapter of the Association for Computing Machinery
Attn.: PGSI
1225 Digital Computer Laboratory, MC 258
1304 West Springfield Ave.
Urbana, IL 61801

14 Acknowledgments

Throughout the development of this product, a number of people contributed in its design, development, and manufacture. These people will never receive any monetary compensation for their efforts, so ACM at UIUC wishes to strongly congratulate the following people for their contributions to this project:

Teresa Johnson, now graduated with a degree in Electrical Engineering, for writing the serial interrupt code used for queued serial data transfer on the PGSI.

Laura Kalman, now graduated with a degree in Electrical Engineering, who served as member-at-large with the development team and provided ideas and scope to the design.

Greg Newby, an assistant professor in the Graduate College of Library and Information Sciences, who served as liaison between Ben Gross and the original PGSA designers (AGE), proposed the initial project to Ben, and shaped the design and manufacture of the PGSI.

Mike Schaffstein, now graduated with a degree in Computer Engineering, who chaired the group that spawned this project, helped construct prototypes, and served as liaison between the design team and the ACM general membership.

Mike Stangel, a senior in Computer Engineering and Chair of ACM at UIUC, who utilized every opportunity to publicize the product and set tentative timetables for development.

Jonathan Stark, a sophomore in Electrical Engineering, for work on some later code development and two glove support.

Chris Wilson, now graduated with a degree in Computer Science, who both tested versions of the code in the prototypes and wrote initial demonstration programs such as Maestro.

15 References

H. Eglowstein. Reach out and Touch your Data. In Byte, pages 283-290, July 1990.